

The Architecture of a Quantum Programming Environment

by

Shusen Liu

A thesis submitted in partial fulfilment of the
requirements for the degree of Doctor of Philosophy

Supervisor: Prof. Runyao Duan

Co-supervisor: Prof. Mingsheng Ying

at the

Centre for Quantum Software and Information
Faculty of Engineering and Information Technology
University of Technology Sydney

October 2018

Certificate of Original Authorship

I, *Shusen Liu* declare that this thesis, is submitted in fulfilment of the requirements for the award of *Doctor of Philosophy Degree*, in *Software Engineering at the Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This thesis is the result of a research candidature conducted jointly with Sun Yat-sen University as part of a collaborative doctoral degree.

This document has not been submitted for qualifications at any other academic institution.

Signed:

Production Note:

Signature removed prior to publication.

Date:

October 2018

The Architecture of a Quantum Programming Environment

by

Shusen Liu

A thesis submitted in partial fulfilment of the requirements for the
degree of Doctor of Philosophy

Abstract

This thesis presents the architecture of quantum programming environment, called QSI, along with its related modules and several quantum experiments. The environment is based on one specific quantum language, namely quantum **while**-language. Some partial experimental results are also presented within QSI.

The first part relates to the architecture, the designing and the implementation of quantum programming environment which provides a new, powerful and flexible environment for developing and implementing quantum programs. First, we study the possible structure of the programming environment which supports a measurement-based case statement and a measurement-based **while**-loop. These two program constructs are extremely convenient for describing large-scale quantum algorithms, such as quantum random walk-based algorithms. We also define a new assembly language called f-QASM (Quantum Assembly Language with feedback) as an interactive command set. The assembly language is compatible with other low-level instruction sets and can be used to directly drive quantum hardware. Moreover, the simulation of syntax of quantum program and the behaviours within the architecture on the classical computer are discussed. Finally, we consider the work-flow which contains the decomposition of unitary matrix to achieve the goal that executing on Noisy Intermediate-Scale Quantum Computer.

The second part concerns the modules based on quantum programming environment: termination analysis module, detective separable unitary module and quantum control module. Along with the architecture, we bring an essential module - termination analysis

module for the loop structure. It can analyze sub-bodies of quantum program and suggest the critical termination information. In addition, we improve the Jordan decomposition step in the original algorithm which consumes extended period for analyzing. This improvement also makes the module more robust on executing. A fast permutation algorithm module clarifies the re-ordering algorithm in case of qubits system. It regenerates the program (unitary operator) which is not in pre-ordered sequence. In the detective separable unitary module, we prove sufficient conditions for separable unitary and its approximate scenario. The result shows there does not exist a universal algorithm for potential parallel executing quantum programs without communications (classical or quantum communications). However, in approximate, there exists a scheme for parallel computing without the help of communication. In this part, two examples for parallel computing are given. Last, in quantum control module, an algorithm is suggested towards automatically generating quantum circuits for quantum case-statement. We believe these analysis modules can help the compiler to optimize the implementation of quantum algorithms.

The third part is devoted to quantum experiment. First, we focus several experiments which can be operated directly by QSI : Qloop, BB84 protocol and Grover search algorithm. After that, with the help of IBM's QISKit, two impressive experiments: distinguishing unitary gates and Bell states are given on real quantum computer. Finally, we combine QSI with Microsoft's LIQUi|) to implement quantum case-statement. These experiments significantly show the quantum power and the scalable framework of the quantum programming environment in practice.

Acknowledgements

The research of this dissertation could not have been performed if not for the assistance, patience, and support of many individuals. First, I would like to express my sincere gratitude to my supervisor Prof. Runyao Duan for supervising my research. His insightful and rigorous instruction has motivated me to constantly pursue studies in the quantum area. His encouragement to not give up when the enormous scope of QSI become too overwhelming was my last resource for achieving this goal. It sincerely touched me that he even contributed an amount some of his own time to debugging and the details of the designing. Also, I wish to thank Prof. Mingsheng Ying for introducing me to the topic and for his support on the way. His knowledge and understanding broadened my views on quantum programming and guided me in the right direction.

I would additionally extend my deepest gratitude to Mr Yang He for sharing his wealth of experience in classical programming with me and saving me with some programming techniques. Without the wisdom and skills he has accumulated over the last two decades, our programming environment would not have been developed so quickly.

I also want to show my thanks to Prof. Kan He. The routine discussions during his visit to Sydney illuminated my thinking on fundamental knowledge and made the ‘impossible’ task of designing the compiler possible.

I also express my thanks to Jemima Moore, who spends much time on proofreading this thesis after final examination. She examines the entire thesis carefully and thoroughly and gives many useful suggestions.

Further, I wish to thank my parents for supporting me as a higher degree research student. They provided me with all their love and care and asked for nothing in return.

Finally, and in particular, I wish to extend my thanks to my love, Ms Huanhua Zhang. She has spent two years with me and fully devoted herself to supporting my research.

Contents

Declaration of Authorship	iii
Abstract	v
Acknowledgements	vii
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Thesis	1
1.2 Background	2
1.3 Aims, Objectives and Significance	3
1.3.1 Aims	3
1.3.2 Objectives	4
1.3.3 Significance	5
1.4 Research Methods	7
1.5 Literature Review	8
1.5.1 Programming Languages and Platforms	8
1.5.2 Quantum Circuits Synthesis and Decompostion	9
1.5.3 Termination Analysis	10
1.5.4 Unitary Discrimination	11
1.5.5 Quantum Control	13
1.6 Preliminaries	14
1.6.1 Quantum while -language	14
1.6.2 QuGCL	15
1.7 Overview	18
2 Quantum Programming Environment	21
2.1 Introduction	21
2.2 The Structure of QSI	22
2.2.1 Basic Features of QSI	22
2.2.2 Main Components of QSI	24
2.2.3 Implementation of QSI	25

2.3	The Quantum Compiler	27
2.3.1	f-QASM	27
2.3.2	Basic Definitions in f-QASM	28
2.3.3	f-QASM Examples	29
2.3.4	Decomposition of a General Unitary Transformation	32
2.4	The Quantum Simulator	33
2.4.1	Quantum Types	33
2.4.2	Simulating of Quantum Behaviors	35
2.4.3	Simulating Operational Semantics in the Quantum while -language	36
3	QSI Modules	41
3.1	Separation of Multipartite Quantum Gates	41
3.1.1	Introduction for Separability of Multipartite Quantum Gates	41
3.1.2	Theoretical Analysis of Quantum Gates Separation	42
3.1.3	Approximate Separation of Multipartite Gates	52
3.1.4	Approximate Separation in a 2-qubit $\frac{1}{2}$ -spin System	55
3.1.5	Conclusion and Discussion	58
3.2	Fast Permutation and its Application in Compiler Module	59
3.2.1	Introduction for Permutation	59
3.2.2	General Algorithm for Permutation	60
3.2.3	A Fast Permutation Algorithm Based on Fixed and Ordered Basis	60
3.3	Termination Analysis Module	62
3.3.1	Introduction for Termination Module	62
3.3.2	Definitions and Theorems	62
3.3.3	Algorithms for Termination Analysis	67
3.3.4	Termination Examples- Qloop	69
3.3.5	Conclusion and Discussion	70
3.4	Quantum Control Module	71
3.4.1	Introduction	71
3.4.2	QMUX Preliminaries	72
3.4.3	Solving Equations	74
4	Experiments	77
4.1	Experiments with QSI	77
4.1.1	Configuration	77
4.1.2	Experiments	79
4.1.3	Serious Coding	96
4.2	Experiments with QISKit and QSI	107
4.2.1	Implementing the Perfect Discrimination of Unitary Operators on IBMQ Cloud Quantum Computer	107
4.2.2	Distinguishing Bell States with Local Measurement on IBMQ	115
4.3	Experiments with QISKit and LIQUi ⟩	119
4.3.1	Typical Cases	119
4.3.2	Extended 2-qubit QMUX	121
5	Summary	123
5.1	Summary of Contributions	123
5.2	Future work	124

Appendices	127
-------------------	------------

Bibliography	127
---------------------	------------

List of Figures

2.1	Framework of QSI	23
2.2	QSI The procedure for simulating in the quantum simulation engine	26
2.3	QSI Quantum types layer	33
3.1	Flowchart of a Qloop	69
3.2	Speed of termination analysis algorithms on two programs of different terminating types.	71
3.3	Decomposition of QMUX	73
3.4	QMUX gate rotation decomposition	73
3.5	Decomposition of arbitrary controlled unitary gate	74
4.1	.NET desktop development	78
4.2	MATLAB support assembly	78
4.3	Examples in Project “UnitTest”	79
4.4	CNOT experiment	79
4.5	Termination	81
4.6	Almost-surely termination	81
4.7	Simple BB84 example	82
4.8	Simple BB84 protocol	83
4.9	Multi-clients BB84 console	86
4.10	BB84 with a quantum channel, success	87
4.11	BB84 with a quantum channel, failure	87
4.12	BB84 with statistics and quantum channel	89
4.13	Statistics of success communication via BB84 with channels	90
4.14	Quantum teleportation with f-QASM	92
4.15	Default network structure	92
4.16	Entry to the user code	97
4.17	Teleportation with loops	103
4.18	Compiled file structure	106
4.19	Parallel and sequential discrimination schemes	109
4.20	The quantum circuit (U_p) generates $ \Psi\rangle$ from $ 0\rangle \otimes 0\rangle$	111
4.21	The quantum circuit (U_m) distinguishes $U^{\otimes 2} \Psi\rangle$ and $ \Psi\rangle$	111
4.22	Statistical results in the parallel discrimination experiments.	113
4.23	Statistical results in the sequential discrimination experiments.	113
4.24	The discrimination success probability distributions	114
4.25	Circuits for generating Bell states	117
4.26	Measurement results with two copies of a Bell state	118
4.27	One copy experiment and computational measurement.	118

4.28 Controlled-unitary gate decomposition	120
4.29 Demultiplexing multiplexed rotation gates	121

List of Tables

4.1	Quantum variable types and corresponding initial methods	99
4.2	Quantum variable initialisation matrix form	99
4.3	Ideal measurement results on two copies of a Bell state	117
4.4	Measurement results of controlled-Hadamard gate	120
4.5	Measurement results of controlled-identity gate	120
4.6	Measurement result of controlled-Bit flip gate	121

List of Publications

- [1] Shusen Liu; Yinan Li; Runyao Duan. Distinguishing unitary gates on the ibm quantum processor (accepted as regular paper). *SCIENCE CHINA Information Sciences*, 2018.
- [43] Shusen Liu, Xin Wang, Li Zhou, Ji Guan, Yinan Li, Yang He, Runyao Duan, and Mingsheng Ying. Qsi: a quantum programming environment. *arXiv preprint arXiv:1710.09500*, 2017.
- [3] Shusen Liu, Yang He, and Cai Zhang. Towards automatically construct quantum circuits for quantum programs with quantum control. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5, June 2017. doi: 10.1109/VTC-Spring.2017.8108699.

